"Bearer Access in Wireless Communication Systems

INTRODUCTION

Field of the Invention

The invention relates to access of servers to Wireless Application Protocol (WAP) bearer systems.

At present, it is known to provide an interface, and a development environment for such an interface, which allows access to Short Message Service Centres (SMSCs) via the (Short Message Peer-to-Peer) SMPP protocol. These products are very effective for the particular interfacing involved. However, many developers for environments such as Web server applications are unfamiliar with wireless interfacing functionality. Also, those developers who are expert in this area often prefer to maintain the bearer-side interfacing functionality private and avoid the problems associated with providing backwards and forwards compatibility and ensuring correct use. Such bearer developers also have the problem of testing their interfaces on multiple server platforms.

The invention is directed towards providing a bearer interface layer to address these problems.

SUMMARY OF THE INVENTION

According to the invention, there is provided an interface layer for a wireless server hosting server applications, the interface layer comprising:

a uniform interface to the wireless server,

a bearer module interface for binding bearer modules to provide access of the server applications to bearer systems, and

functions between said layers to allow server applications to make calls to connect to a bearer module and to determine status of accessible bearer systems.

In one embodiment, the functions comprises a configuration manager comprising means for maintaining configuration data relating to bearer modules and for responding to server application calls for a configuration list.

In one embodiment, the configuration manager comprises means for returning information regarding network type, bearer system type, and address type to server applications.

In one embodiment, the configuration manager comprises means for capturing bearer module information configured by a bearer developer within constraints of interfacing to the bearer module interface.

In one embodiment, the configuration manager comprises means for maintaining a configuration file for every bound bearer module.

In one embodiment, the functions include a bearer manager comprising means for registering bearer modules and for providing bearer module status information to requesting server applications.

In one embodiment, the bearer manager comprises means for responding to server application calls to register a bearer module.

In one embodiment, the bearer manager comprises means for making a bind attempt to bind a bearer module to a transport mechanism.

In one embodiment, the bearer manager comprises means for responding to application calls to disconnect from a bearer module.

In one embodiment, the functions include a blacklist and whitelist manager comprising means for automatically testing all non-secure traffic.

In one embodiment, the blacklist and whitelist manager comprises means for filtering traffic according to parameters set by a server application.

In one embodiment, the functions include a callback handler comprising means for allowing a server application to call routines for pre-processing and post-processing of WDP packets.

In one embodiment, the functions include a log function comprising means for logging test data from a bearer module under test, and for providing said data to a requesting server application.

In one embodiment, the functions include a billing function comprising means for generating billing data on the basis of criteria supplied by a server application.

According to another aspect, the invention provides a wireless application server comprising WSP and WTP layers over an interface layer as defined above.

DETAILED DESCRIPTION OF THE INVENTION

Brief Description of the Drawings

The invention will be more clearly understood from the following description of some embodiments thereof, given by way of example only with reference to the accompanying drawings in which:

Fig. 1 is an overview diagram showing the context of an interface layer of the invention;

Fig. 2 is a diagram showing the structure of the interface layer; and

Figs. 3 and 4 are diagrams illustrating uses of the interface layer; and

Fig. 5 is a diagram illustrating signal interactions.

Description of the Embodiments

Referring to Fig. 1, interface layers 1 allow access by wireless servers 2 to bearer systems 3. In the embodiment illustrated, the servers 2 include:-

- a WAP server 5 hosting prototype application and content,
- a WAP proxy 6 performing protocol conversion and content encoding, and
- a WTA server 7 hosting telephony based applications.

The servers are connected to a Web server 8 via a Firewall 9. The bearer systems are in this embodiment:-

- a Circuit Switched Data bearer system 10,
- an Express Messaging SMSC bearer system 11,
- an SMSC bearer system 12, and
- a USSD gateway bearer system 13.

Referring to Fig. 2 each interface layer 1 comprises a server interface 10 to the applications and WTP (Wireless Transaction Protocol) layer of the server. The interface 10 is uniform in its interfacing operations to all of the server applications. It also comprises an interface 11 to bearer system interfacing modules 20, called "bearers" or "bearer modules" in this specification. Between these two interfaces there is:

a bearer manager 12,
a configuration manager 13,
a log 14,
a callback handler 15,
a blacklist/whitelist manager 16,
a billing and CDR function 17, and
an error function 18.

The interface 10 is a uniform interface to all of the server applications and the interface 11 allows bearers 20 to be loaded and offers their services to the applications.

Referring to Fig. 3, one of the bearers 20 is a test bearer which writes to the log 14 and the log 14 returns responses to an application 25 on a WAP stack 26. This is a convenient framework for creation of new bearer modules 20. It also serves to examine and explain the facilities offered by an existing bearer 20. It is thus very suitable for testing bearer modules 20 and allows short development times.

Referring to Fig. 4, a WAP gateway 40 having a WAP stack 41 uses the interface layer 1 to access a data gateway 42 and an SMSC 43. Each of these bearer systems is accessed via a bearer module 20 (Bearer A and Bearer B). However the gateway 42 is also accessed by a private interface 44 to be kept secret, while the SMSC is

accessed via SMPP 45 (which is public). It will thus be appreciated that the layer 1 allows versatility in the manner in which bearer developers may link their bearer system with a server: a private or a public protocol may be used. All of the bearer modules 20 are bound by the rules of the interface 11, but may or may not be private on the bearer system side. This is illustrated in Fig. 4 in which there is a secret interface 46 and an SMPP interface 47 on the bearer system side. Each bearer module 20 consists of one or more bearer files and bearer configuration files. Where possible, bearer modules are designed for dynamic loading but where this is not possible or not desirable static linking of bearer modules is supported. The following describes the functions between the interfaces 10 and 11 in more detail.

Configuration manager 13

The manager 13 maintains a file which specifies which bearer modules are present, and there is a configuration file of bearer module parameters for each bearer module.

Bearer manager 12

The bearer manager 12 is responsible for the loading and management of the bearer modules. It performs version checking and license checking

Billing & CDRs 17

This can be set to be an automatic operation to generate billing information (and pre-paid queries) for all non secure traffic or it can be invoked by the server application for all types of traffic. Billing is on a basis set by the server application, such as per-URL or per data volume.

White & Blacklist Manager 16

This can be set to be in 'automatic operation' and automatically test all non-secure traffic, or can be invoked by a server application for required types of traffic. The contents of the lists are URLs or originator addresses.

Callback Handler 15

This allows the server application to request the layer 1 call its routines to provide extended pre and post processing of WDP packets.

The calls available to the server applications are set out in the table below.

wdc_bearer_list	Returns the list of bearer modules 20 configured on the server	
wdc_info	Get information about a specific bearer module.20	
wdc_connect	Connect to a bearer module 20	
wdc_send	Send a message to the specified bearer system	
wdc_get	Get a message from a specified bearer system	
wdc_param	Get details of a configuration parameter from a configuration file	
wdc_log	Create a log entry in the log/trace files.	
wdc_close	Disconnect from a bearer module.	

The following handlers are provided to the bearer developer

wdc_register	Handle the registration of a bearer module.	
wdc_connect	Connect to a bearer module.	
wdc_send	Send a message to the specified bearer	
wdc_get	Get a message from a specified bearer	
wdc_param	Get details of a configuration parameter from a configuration files	
wdc_info	Get information about a specify bearer module.	
wdc_log	Create a log entry in the log/trace files.	
wdc_close	Disconnect from a bearer module.	

Some of the above calls are illustrated in Fig. 5, in which server applications communicate with an SMSC bearer system for bi-directional transfer of WAP content. The following describes the calls in more detail, with examples of code.

wdc_bearer_list

Returns the list of bearer modules configured by the system. The call actually returns the value of the bearer list parameter from the wdc-dtk.ini (layer 1 configuration file).

Each item in the bearer list is known as a bearer tag and can be used to retrieve information on that bearer using the "wdc_info" call.

wdc-dtk.ini

[main]

Bearer_List = SMSC01, SMSC_DG, USSD, CSD23, EXP-001

Each of the items in the list will have their own section in the ".ini" file.

[SMSC01]

bearer_modue = "smsc01/smsc3-4"

parameter_file = "smsc01/smsc.ini"

network = "GSM"

bearer_type = "SMS"

address_type = "GSM_MSISDN"

assigned_number = "0x03"

wdc_info

Returns an INFO element for a connection. The information returned includes the information from the "Network Bearer table" defined by WAP.

- Network
- Bearer type
- Address Type
- Assigned Number

A connection is not required to obtain information about the bearer.

When a new bearer module 20 is being created a handler for each of the above calls is provided. How the call is implemented is decided by the developer of each bearer module within the constraints imposed by the interface 11. For example a developer

can choose to implement the "wdc_parm" handler to read from a flat file, to read from a database or to permanently return a blank value.

wdc_register

This is called by the application to load/register a specific bearer module. One of the parameters passed is the version identifier of the interface to ensure that only modules of a correct or later version are loaded. If the layer 1 locates an older version of the bearer module then that which it expects an error is returned.

wdc_connect

This function is used by the application to connect to a bearer. It makes an application layer connection to the bearer. The bearer module should, on a 'bind', attempt to connect/bind to the transport mechanism to ensure it is in place. If an application wants to connect to multiple bearer systems, say USSD and SMS, then it must make multiple bind calls:

socket = wdc_connect(bearer);
e.g.
socketSMS = wdc_connect(GSM_SMS);

Returns
Success: This function returns a non-NULL connection identifier which will be used by all further calls to this bearer until the connection is terminated.

Failure: The function returns null and stores an error in the global error number variable.

wdc_close

This tells the bearer module that the server application is disconnecting. All resources allocated to using the bearer should be freed.

```
wdc_close(socketSMS);
```

wdc_send

Is used to send information to a data gateway via a bearer module using the layer 1.

wdc_get

Is used to read information from a data gateway via a bearer module using the layer 1.

wdc_log

```
wdc_log(socket, severity, msg);
```

An example is:

```
wdc_log(socketSMS, WARNING, "Less the 10 % Memory remaining.");
```

Regarding billing and prepaid queries, one of the areas the layer 1 simplifies is the complex area of interfacing to telecom billing and prepaid systems. It is possible for an application to instruct the layer 1 to load and enable a billing sub module. The application may chose to enable the layer 1 to automatically log location URL and/or Volume of data as CRD for billing. This is only possible for WAP if the non-secure port is used.

Wdc_cdr_mode(setting)

- VOLUME
- URL
- OFF

The INI setting "billing_module" points to the location of the billing module to load.

The application may call the layer 1 to create CDRs.

wdc_cdr_volume(CDR_HEADER, 400);
Log 400 bytes

wdc_cdr_url(CDR_HEADER, "http://www.wapforum.com/index.wml");

Other calls include Whitelist and Blacklist management, in which an application may enable automatic Black/White list checking. The application may call the layer 1 to check a URL against the Black/While list using :

wdc_check_mode(ON/OFF);

wdc_check_url(check_interface, url);

The INI parameter:

"Black_while_list" indicates the file to use.

wdc_parm

This is used by applications for parameter reading:
err_code_t wdc_parm(bearer_tag, wdc_param_t, *parameter);

```
typedef struct
{
        char name[WDC_PARAM_NAME_SIZE];
        char value[WDC_PARAM_VALUE_SIZE];
```

} wdc_pararm_t;

This call is used to get a value of a configuration parameter from one of the configuration files. The wdc_module parameter selects which of the module – MAIN or one of the bearers is queried. Having a separate configuration file for each bearer module is necessary to prevent naming conflicts when bearer modules are provided by third parties.

RETURNS

Success: The function returns and ERR_OK and the value of the parameter is placed in the value field of the structure.

Failure: The function returns a non-zero error number.

Callbacks are provided by the layer 1 to allow an application request that it handles various events:

wdc_callback(item, handler)

ERROR – allows the application register its own handler for error conditions this is useful it the application already has an error logging facility that it wishes to use.

LOG – allow an application to register its own handler for logging messages.

It will be appreciated that the invention provides for excellent versatility for both bearer-side and server-side development. The interface layer 1 also allows the addition of new bearers as they become available, and the application to utilise 'intelligent' selection of bearers.

The invention is not limited to the embodiments described, but may be varied in construction and detail.